

# Polynomial-time kernel reductions

Jeffrey Finkelstein  
Boston University  
<jeffreyf@bu.edu>

Benjamin Hescott  
Tufts University  
<hescott@cs.tufts.edu>

April 29, 2016

## Abstract

Today, the computational complexity of equivalence problems such as the graph isomorphism problem and the Boolean formula equivalence problem remain only partially understood. One of the most important tools for determining the (relative) difficulty of a computational problem is the many-one reduction, which provides a way to encode an instance of one problem into an instance of another. In equivalence problems, the goal is to determine if a pair of strings is related, so a many-one reduction with access to the entire pair may be too powerful. A recently introduced type of reduction, the *kernel reduction*, defined only on equivalence problems, allows the transformation of each string in the pair independently. Understanding the limitations of the kernel reduction as compared with the many-one reduction improves our understanding of the limitations of computers in solving problems of equivalence. We investigate not only these limitations, but also whether classes of equivalence problems have complete problems under kernel reductions. This paper provides a detailed collection of results about kernel reductions.

After exploring possible definitions of complexity classes of equivalence relations, we prove that polynomial time kernel reductions are strictly less powerful than polynomial time many-one reductions. We also provide sufficient conditions for complete problems under kernel reductions, show that completeness under kernel reductions can sometimes imply completeness under many-one reductions, and finally prove that equivalence problems of intermediate difficulty can exist under the right conditions. Though kernel reductions share some basic properties with many-one reductions, ultimately the number and size of equivalence classes can prevent the existence of a kernel reduction, regardless of the complexity of the equivalence problem. The most important open problem we leave unsolved is proving the unconditional existence of a complete problem under kernel reductions for some basic complexity classes that are well-known to have complete problems under many-one reductions.

---

Copyright 2010–2016 Jeffrey Finkelstein and Benjamin Hescott.

This document is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License, which is available at <https://creativecommons.org/licenses/by-sa/4.0/>.

# 1 Introduction

The computational complexity of deciding whether two graphs are isomorphic has significant implications not only in computer science, but also in the computational forms of sciences such as chemistry, biology, and neuroscience. One main technique for determining the complexity of the problem is showing how the difficulty of the problem relates to the difficulty of other known problems. The relative difficulty of computational problems are often compared using the many-one reduction, a function by which we encode an instance of a problem as an instance of another problem. In the case of the graph isomorphism problem, a many-one reduction from the graph isomorphism problem to, for example, the directed graph isomorphism problem allows the function computing the reduction to have access to both graphs in an instance of the problem. However, access to both graphs is not necessary for computing the reduction; the function transforms each undirected graph independently into a directed graph. In other words, the reduction is in reality defined on the domain of graphs, not on the domain of pairs of graphs. This is a far more natural way to define reductions between problems of equivalence, and is furthermore a finer-grained comparison of the relative difficulty of the two computational problems.

The *kernel reduction*, defined in [12, Definition 4.13], formally captures this notion of reduction among computational problems of equivalence involving independent transformation of each element of a pair. This type of reduction has appeared previously under other names not only in this setting but also in more general settings (“Borel reduction,” “strong isomorphism reduction,” “strong equivalence reduction,” “relation reduction,” “component-wise reduction,” etc.). To the best of our knowledge, every known many-one reduction between problems of equivalence is really a kernel reduction (see, for an early example, the list of problems many-one reducible to graph isomorphism given in [5]). However, kernel reductions seem less powerful than many-one reductions, since the former has access only to one element of a pair at a time. What are the limitations of kernel reductions?

Some of our theorems adapt or clarify existing work in order to have simpler, self-contained, complexity-theoretic proofs of important theorems about kernel reductions. In [12], the authors ask whether kernel reductions and many-one reductions are provably different. However, little beyond the definition is given there, other than the general idea that an imbalance in the number of equivalence classes of the two equivalence problems prevents the existence of a kernel reduction. In computability theory, a similar type of reduction between equivalence problems has been well-studied by a series of recent papers (for example, [13, 11, 10, 8, 16, 1, 20]). However, these papers do not focus on efficiently computable reductions. In [6], the authors provide a thorough treatment of not only the kernel reduction but also a generalization called the “strong isomorphism reduction.” Strong isomorphism reductions are themselves a special case of “functorial

---

The L<sup>A</sup>T<sub>E</sub>X markup that generated this document can be downloaded from its website at <https://github.com/jfinkels/equivalence>. The markup is distributed under the same license.

reductions” (a name borrowed from the language of category theory), which are reductions that make explicit the morphism between the category of objects being transformed. These reductions were apparently defined in unpublished manuscripts [3] and [17] (see [24, Section 15] for a contemporary definition, [2, Section 7] for a more recent one). Finally, the authors of [14] extended the work of [6], and in doing so, independently proved the main combinatorial idea used in this paper to examine the limitations of kernel reductions. This paper, complementing that work, focuses mainly on completeness results.

We undertake a thorough investigation of the basic properties of kernel reductions, comparing them with the basic properties of many-one reductions. The starting point for understanding many-one reductions is  $P$  and  $NP$ , so we attempt to extend the definition from [12] of  $PEq$ , the class of equivalence problems decidable in polynomial time, to the definition of the complexity class  $NPEq$  (section 3). We determine the limitations of kernel reductions; these appear to be combinatorial, not computational, in nature (section 4). We discover sufficient conditions for complete problems under kernel reductions in classes of equivalence problems (section 5). We compare the new notion of completeness under kernel reductions with the usual notion of completeness under many-one reductions (section 6). Finally, as an analog to  $NP$ -intermediary problems with respect to many-one reductions, we examine the possibility of  $NPEq$ -intermediary problems with respect to kernel reductions (section 7).

## 2 Preliminaries

The set of natural numbers (including 0) is denoted  $\mathbb{N}$ , the set of integers is denoted  $\mathbb{Z}$ , and the set of positive integers is denoted  $\mathbb{Z}^+$ .

If  $f: S \rightarrow T$  is a well-defined function and  $S' \subseteq S$ , then  $f$  *restricted to the domain*  $S'$  is the function  $f': S' \rightarrow T$  defined by  $f'(x) = f(x)$  for all  $x \in S'$ . We denote this restricted function on a smaller domain by  $f|_{S'}$ . The *image of*  $S'$ , denoted  $f(S')$ , is defined by  $f(S') = \{f(s) \mid s \in S'\}$ .

In this paper,  $\Sigma$  denotes the binary alphabet  $\{0, 1\}$ .  $\Sigma^*$  is the set of all binary strings over the alphabet  $\Sigma$  and  $\Sigma^{\leq n}$  is the set  $\{w \in \Sigma^* \mid |w| \leq n\}$ . The empty string will be denoted by  $\lambda$ . If  $\sigma \in \Sigma$  then  $\sigma^k$  is the string consisting of  $k$  concatenated copies of the symbol  $\sigma$ . If  $x$  and  $y$  are elements of  $\Sigma^*$ , then we denote by  $\langle x, y \rangle$  the *pairwise encoding* of  $x$  and  $y$ , which is itself an element of  $\Sigma^*$ . In this paper, we will assume the reasonable pairwise encoding defined by  $\langle x, y \rangle = x_1x_1x_2x_2 \cdots x_{|x|}x_{|x|}01y_1y_1y_2y_2 \cdots y_{|y|}y_{|y|}$  for all  $x$  and  $y$  in  $\Sigma^*$ . As usual, a *language* over an alphabet  $\Sigma$  is a subset of  $\Sigma^*$ . The *complement* of a language  $L$  is  $\Sigma^* \setminus L$ , and is denoted  $\bar{L}$ .

The complexity classes  $P$ ,  $NP$ ,  $FP$  (polynomial-time computable functions),  $\Sigma_k P$ ,  $\Pi_k P$ ,  $\Delta_k P$ , and  $PSPACE$  have the usual definitions. The set of words accepted by a Turing machine  $M$  is denoted  $L(M)$ . The *complement* of a complexity class  $\mathcal{C}$  is the set of complements of languages in  $\mathcal{C}$ , and is denoted  $\text{co}\mathcal{C}$ .

We say a Turing machine  $M$  is a *polynomially clocked Turing machine* if the

description of  $M$  includes a positive integer  $k$  such that  $M$  halts within time  $kn^k$  on all inputs of length  $n$ .

If  $L_1$  and  $L_2$  are languages, we say that  $L_1$  *many-one reduces to*  $L_2$  if there exists a computable function  $f$  such that  $w \in L_1$  if and only if  $f(w) \in L_2$ . We denote this by  $L_1 \leq_m L_2$ . If  $f$  is computable in polynomial time, we denote this by  $L_1 \leq_m^P L_2$ .

A set  $R \subseteq \Sigma^* \times \Sigma^*$  is an *equivalence relation on  $\Sigma^*$*  if  $R$  satisfies the following three properties.

- (reflexivity) For all  $x \in \Sigma^*$ ,  $(x, x) \in R$ .
- (symmetry) For all  $x, y \in \Sigma^*$ ,  $(x, y) \in R$  implies  $(y, x) \in R$ .
- (transitivity) For all  $x, y, z \in \Sigma^*$ ,  $(x, y) \in R$  and  $(y, z) \in R$  implies  $(x, z) \in R$ .

An equivalence relation  $R$  can be encoded as a language by taking the pairwise encoding of each pair in  $R$ . In this way we can study the computational complexity of classes of languages which represent equivalence relations. In this paper we will abuse notation and write  $\langle x, y \rangle \in R$  for an equivalence relation  $R$  on  $\Sigma^*$ , but what we really mean is  $(x, y) \in R$  and  $\langle x, y \rangle \in L_R$ , the language on the alphabet  $\Sigma$  induced by  $R$ .

The *equivalence class* of  $x$  with respect to an equivalence relation  $R$  on  $\Sigma^*$  is  $\{y \in \Sigma^* \mid (x, y) \in R\}$ . It is denoted  $[x]_R$ , or if the context is clear, simply  $[x]$ . Each element  $x \in \Sigma^*$  is in exactly one equivalence class, so the equivalence classes of an equivalence relation on  $\Sigma^*$  provide a partition of  $\Sigma^*$ . Conversely, a partition of  $\Sigma^*$  induces an equivalence relation on  $\Sigma^*$  in which a pair of elements is in the relation if they are in the same block of the partition.

A *complete invariant* for an equivalence relation  $R$  on  $\Sigma^*$  is a function  $f: \Sigma^* \rightarrow \Sigma^*$  such that for each  $x$  and  $y$  in  $\Sigma^*$ , we have  $(x, y) \in R$  if and only if  $f(x) = f(y)$ . (A *canonical form* for an equivalence relation is a complete invariant satisfying the additional requirement that  $f(x) \in [x]_R$ ; canonical forms, though important, do not appear in this paper.) In section 3 we will define generalizations of the complete invariant which accept as input an additional witness to the equivalence of  $x$  and  $y$ .

**PEq** is the class of equivalence relations for which membership can be decided by a Turing machine running in deterministic polynomial time. **NPEq** is the class of equivalence relations for which membership can be decided by a Turing machine running in non-deterministic polynomial time. In other words, **PEq** is the set of (languages induced by) equivalence relations which are in **P**, and **NPEq** is the set of (languages induced by) equivalence relations which are in **NP**. In general, the class **CEq** is the class of languages induced by equivalence relations which are in the complexity class  $\mathcal{C}$ . As usual,  $\mathbf{PEq} \subseteq \mathbf{NPEq}$ .

We now require a natural notion of reduction among equivalence relations. If  $R$  and  $S$  are equivalence relations on  $\Sigma^*$ , we say  $R$  *kernel reduces to*  $S$  if there exists a computable  $f: \Sigma^* \rightarrow \Sigma^*$  such that  $\forall x, y \in \Sigma^*$ ,  $\langle x, y \rangle \in R \iff \langle f(x), f(y) \rangle \in S$ . We denote this by  $R \leq_{ker} S$ . If  $f$  is computable in polynomial

time, then we say  $R$  *polynomial-time kernel reduces to*  $S$  and use the notation  $R \leq_{ker}^P S$ .

Notice the difference between a kernel reduction and a many-one reduction: a kernel reduction maps  $\langle x, y \rangle \in R$  to  $\langle f(x), f(y) \rangle \in S$ , whereas a many-one reduction maps  $\langle x, y \rangle \in R$  to  $f(\langle x, y \rangle) \in S$ , for some polynomial-time computable function  $f$ . Informally, a function which computes a many-one reduction has access to both  $x$  and  $y$  but a function which computes a kernel reduction has access to only one of  $x$  and  $y$  at a time. Since it is more restrictive, a kernel reduction induces a many-one reduction (namely the function  $\langle x, y \rangle \mapsto \langle f(x), f(y) \rangle$ ). Still, kernel reductions compose just as many-one reductions do, and **NPEq** is closed under polynomial-time kernel reductions, allowing us to adapt existing complexity theoretic analysis to the study of complexity of equivalence relations.

As an analog to polynomial-time many-one completeness in **NP**, we define a similar notion of completeness under polynomial-time kernel reductions in **NPEq**. An equivalence relation  $S$  is **NPEq-hard** if for all  $R \in \mathbf{NPEq}$ ,  $R \leq_{ker}^P S$ . If  $S$  is also in **NPEq**, then it is **NPEq-complete**. If  $S$  is **NPEq-complete**, we sometimes say that  $S$  is *complete under  $\leq_{ker}^P$  reductions in **NPEq***. Generally, an equivalence relation  $S$  is **CEq-hard** if for all  $R \in \mathbf{CEq}$ ,  $R \leq_{ker}^P S$ , and **CEq-complete** if it is additionally in **CEq**.

### 3 Definitions of **NPEq**

The main property of languages in **NP** is that membership in each language is verifiable in polynomial time, given a witness to the membership. Many important equivalence problems are in **NP**, and some are even **NP-complete**, but these are complete under traditional many-one reductions, not kernel reductions. We wish to define **NPEq** as the class of equivalence problems that are efficiently verifiable, just as we define **NP** as the class of all computational problems. One way to define **NPEq** is simply as the subclass of **NP** that includes only equivalence problems. This section provides some other possible definitions based on our intuition about “efficiently verifiable” equivalence problems and compares those definitions.

We show that the alternative definitions of **NPEq** form a hierarchy below **NPEq** as defined above. In other words, **NPEq** is the most general class of efficiently verifiable equivalence problems. When attempting to prove that there are complete problems in **NPEq** under kernel reductions, we must therefore use this most general definition. It remains to show whether any of the (non-equal) alternative definitions are distinct, and whether any of them has a complete problem under kernel reductions.

The first definition is the analog of the fundamental definition of **NP**; it is the formal definition of the class **NPEq** introduced in the previous section.

**Definition 3.1.** An equivalence relation  $R$  is in **NPEq** if there is a polynomial  $p$  and a nondeterministic Turing machine  $N$  such that for each  $x$  and  $y$ , the

machine  $N$  halts in time  $p(|\langle x, y \rangle|)$  and

$$\langle x, y \rangle \in R \iff N(\langle x, y \rangle) \text{ accepts.}$$

Just as there is a definition of **NP** using polynomial-time verifiers, there is an equivalent definition for **NPEq** using polynomial-time verifiers. However, this definition feels a bit unnatural when dealing with equivalence relations, since the witness language would be a relation (of the form “ $(x, y)$  relates to  $w$ ”), but not an *equivalence* relation. The next two definitions attempt to require that the witness language is itself an equivalence relation, instead of an arbitrary language in **P**. Each of these “witness equivalence relations” is a set of pairs of pairs, in which each inner pair includes a witness string.

**Definition 3.2.** Suppose  $R'$  is an equivalence relation in **PEq**. An equivalence relation  $R$  is a *two-witness projection* of  $R'$  if for each binary string  $x$  and  $y$ ,

$$\langle x, y \rangle \in R \iff \exists w_x, w_y: \langle \langle x, w_x \rangle, \langle y, w_y \rangle \rangle \in R',$$

where  $|w_x|$  is polynomially bounded in  $|x|$  and  $|w_y|$  is polynomially bounded in  $|y|$ . The class **Proj<sub>2</sub>** is the collection of all two-witness projections of equivalence relations in **PEq**.

**Definition 3.3.** Suppose  $R'$  is an equivalence relation in **PEq**. An equivalence relation  $R$  is a *one-witness projection* of  $R'$  if for each binary string  $x$  and  $y$ ,

$$\langle x, y \rangle \in R \iff \exists w: \langle \langle x, w \rangle, \langle y, w \rangle \rangle \in R',$$

where  $|w|$  is polynomially bounded in  $\min(|x|, |y|)$ . The class **Proj<sub>1</sub>** is the collection of all one-witness projections of equivalence relations in **PEq**.

The next two definitions attempt to allow the possibility of not just a simple string which witnesses the equivalence of  $x$  and  $y$ , but a “witness function” which may map  $x$  and  $y$ , along with witness strings, to an equivalence relation in **PEq**.

**Definition 3.4.** Suppose  $R$  and  $R'$  are equivalence relations. A function  $f$  is a *nondeterministic polynomial-time two-witness kernel reduction* from  $R$  to  $R'$  if  $f$  is in **FP** and for each binary string  $x$  and  $y$ ,

$$\langle x, y \rangle \in R \iff \exists w_x, w_y: \langle f(x, w_x), f(y, w_y) \rangle \in R',$$

where  $|w_x|$  is polynomially bounded in  $|x|$  and  $|w_y|$  is polynomially bounded in  $|y|$ . The class **Cl<sub>2</sub>** is the closure of **PEq** under these reductions.

**Definition 3.5.** Suppose  $R$  and  $R'$  are equivalence relations. A function  $f$  is a *nondeterministic polynomial-time one-witness kernel reduction* from  $R$  to  $R'$  if  $f$  is in **FP** and for each binary string  $x$  and  $y$ ,

$$\langle x, y \rangle \in R \iff \exists w: \langle f(x, w), f(y, w) \rangle \in R',$$

where  $|w|$  is polynomially bounded in  $\min(|x|, |y|)$ . The class **Cl<sub>1</sub>** is the closure of **PEq** under these reductions.

The final two definitions attempt to describe equivalence relations for which there is a “witnessed complete invariant,” which maps equivalent strings to *equal* strings when given access to some witness of their equivalence.

**Definition 3.6.** Suppose  $R$  is an equivalence relation. A function  $f$  is a *nondeterministic polynomial-time two-witness complete invariant* for  $R$  if  $f$  is in FP and for each binary string  $x$  and  $y$ ,

$$\langle x, y \rangle \in R \iff \exists w_x, w_y: f(x, w_x) = f(y, w_y),$$

where  $|w_x|$  is polynomially bounded in  $|x|$  and  $|w_y|$  is polynomially bounded in  $|y|$ . The class  $\text{NKer}_2$  is the collection of all equivalence relations that admit such a function.

**Definition 3.7.** Suppose  $R$  is an equivalence relation. A function  $f$  is a *nondeterministic polynomial-time one-witness complete invariant* for  $R$  if  $f$  is in FP and for each binary string  $x$  and  $y$ ,

$$\langle x, y \rangle \in R \iff \exists w: f(x, w) = f(y, w),$$

where  $|w|$  is polynomially bounded in  $\min(|x|, |y|)$ . The class  $\text{NKer}_1$  is the collection of all equivalence relations that admit such a function.

The definitions of these complexity classes yield a chain of inclusions beginning with  $\text{NKer}_1$  and terminating with  $\text{NPEq}$ .

**Theorem 3.8.**  $\text{NKer}_1 = \text{Cl}_1 = \text{Proj}_1 \subseteq \text{NKer}_2 \subseteq \text{Cl}_2 = \text{Proj}_2 \subseteq \text{NPEq}$ .

*Proof sketch.*  $\text{Proj}_1 \subseteq \text{Cl}_1$  by choosing the kernel reduction  $f$  to be the identity function.  $\text{Cl}_1 \subseteq \text{NKer}_1$  by choosing the complete invariant  $f'$  to be

$$f'(x, w') = \begin{cases} w'0 & \text{if } \langle f(x, v), f(y, v) \rangle \in R', \text{ where } w' = (y, v) \\ x1 & \text{otherwise,} \end{cases}$$

where  $f$  is the kernel reduction.  $\text{NKer}_1 \subseteq \text{Proj}_1$  by choosing  $R'$  to be the equality relation after an application of the complete invariant  $f$  to both the left pair and the right pair in the relation.

$\text{NKer}_1 \subseteq \text{NKer}_2$  by choosing both  $w_x$  and  $w_y$  to be the witness  $w$ .  $\text{NKer}_2 \subseteq \text{Cl}_2$  by choosing  $R'$  to be the equality relation.

$\text{Proj}_2 \subseteq \text{Cl}_2$  by choosing  $f$  to be the identity function.  $\text{Cl}_2 \subseteq \text{Proj}_2$  by hardcoding the function  $f$  into the relation  $R'$ .

$\text{Proj}_2 \subseteq \text{NPEq}$  by defining  $N$  to nondeterministically choose  $w_x$  and  $w_y$  then verify that  $\langle x, w_x \rangle$  and  $\langle y, w_y \rangle$  are related under  $R'$ .  $\square$

We are unable to show  $\text{Cl}_2 \subseteq \text{NKer}_2$  using the technique that shows  $\text{Cl}_1 \subseteq \text{NKer}_1$  because the complete invariant  $f'$  cannot access both of the necessary witnesses for the kernel reduction  $f$  in a symmetric way. The best we can do is show this inclusion under an assumption.

Our one-witness and two-witness complete invariants are generalizations of the deterministic complete invariant, as defined in section 2. In [12], the authors define the class  $\text{Ker}$  as the set of all equivalence relations  $R$  that have a polynomial-time computable complete invariant. They provide evidence that  $\text{Ker}$  and  $\text{PEq}$  are different by showing that equality of the two classes implies some unlikely collapses in “higher” complexity classes. Unfortunately, we are only able to show that  $\text{Cl}_2 \subseteq \text{NKer}_2$  under the assumption that  $\text{Ker} = \text{PEq}$ .

**Corollary 3.9.** *If  $\text{Ker} = \text{PEq}$ , then  $\text{NKer}_2 = \text{Cl}_2 = \text{Proj}_2$ .*

*Proof.* By the previous theorem, it suffices to show  $\text{Proj}_2 \subseteq \text{NKer}_2$ . Suppose  $R \in \text{Proj}_2$ , so there is an  $R' \in \text{PEq}$  such that  $\langle x, y \rangle \in R$  if and only if there are  $w_x$  and  $w_y$  such that  $\langle \langle x, w_x \rangle, \langle y, w_y \rangle \rangle \in R'$ . Since  $\text{Ker} = \text{PEq}$ , there is a function  $f \in \text{FP}$  such that  $\langle \langle x, w_x \rangle, \langle y, w_y \rangle \rangle \in R'$  if and only if  $f(x, w_x) = f(y, w_y)$ . Thus there is a function  $f$  such that  $\langle x, y \rangle \in R$  if and only if there are  $w_x$  and  $w_y$  such that  $f(x, w_x) = f(y, w_y)$ . Therefore  $R \in \text{NKer}_2$ .  $\square$

## 4 Limitations of kernel reductions

Can a kernel reduction be used anywhere a many-one reduction can be used? If so, a function with access to one element of a pair would be exactly as powerful as a function with access to both elements of a pair; our intuition is that this is unlikely. This section proves that polynomial-time kernel reductions are strictly weaker than polynomial-time many-one reductions.

We find that a bound on the size of the image of a kernel reduction implies that the function can only access a finite number of equivalence classes. Constructing equivalence relations so that there is an imbalance in the number of equivalence classes with respect to any fixed function suffices to show that no polynomial-time kernel reduction can exist between the two. Thus, we conclude that polynomial-time kernel reductions are more restrictive than polynomial-time many-one reductions. This will be important for section 5 as well, since it means that completeness under kernel reductions is distinct from completeness under many-one reductions.

We adopt and extend the notation  $\#R$  from [6] to denote the number of equivalence classes in an equivalence relation  $R$ .

**Definition 4.1** ([6, Section 5]). Suppose  $R$  is an equivalence relation on  $\Sigma^*$ . Let  $\#R(n) = |\{[x]_R \mid x \in \Sigma^{\leq n}\}|$ , or in other words,  $\#R(n)$  is the number of equivalence classes in  $R$  for strings of length at most  $n$ . Let  $\#R = \max_{n \in \mathbb{N}} \#R(n)$  if the maximum exists, or in other words,  $\#R$  is the number of equivalence classes in  $R$ .

As first stated in [12], if the number of equivalence classes in  $R$  is greater than the number of equivalence classes in  $S$ , then no kernel reduction can exist (regardless of any time or space bounds on the function computing the reduction). For completeness, we prove this basic fact in Proposition 4.3 below. However, a



many-one reduction can overcome this restriction by having access to both strings in the pair. Before proving that, we require the following lemma showing that kernel reductions must preserve “related-ness” of pairs of elements by mapping equivalence classes in  $R$  to equivalence classes in  $S$ . (The proof, omitted here, is a straightforward application of the definitions.)

**Lemma 4.2.** *Suppose  $R$  and  $S$  are equivalence relations on  $\Sigma^*$ . Suppose  $R \leq_{ker} S$  and  $f$  is the function computing the kernel reduction. Let  $\hat{f}$  denote the function defined by  $\hat{f}([x]_R) = [f(x)]_S$ , for all equivalence classes  $[x]_R$  in  $R$ . Then*

- $\hat{f}$  is injective,
- $f([w]_R) \subseteq \hat{f}([w]_R)$  for any  $w \in \Sigma^*$ .

**Proposition 4.3.** *Let  $R$  and  $S$  be equivalence relations on  $\Sigma^*$ . If  $\#R > \#S$ , then  $R \not\leq_{ker} S$ .*

*Furthermore, suppose  $\#R = n$  and  $\#S = m$ , and suppose  $m \geq 2$ . Let  $r_1, \dots, r_n$  and  $s_1, \dots, s_m$  denote representatives of the equivalence classes in  $R$  and  $S$ , respectively. If the problem of deciding whether  $x \in [r_i]_R$  for any  $x \in \Sigma^*$  is recognizable, then  $R \leq_m S$ .*

*Proof.* Assume that  $R \leq_{ker} S$ . By Lemma 4.2, the function mapping equivalence classes in  $R$  to equivalence classes in  $S$  induced by the kernel reduction is injective. However, this violates the pigeonhole principle. Therefore no kernel reduction exists from  $R$  to  $S$ .

On the other hand, there is a many-one reduction from  $R$  to  $S$ . First, suppose  $S$  has  $m$  equivalence classes and let  $s_1, \dots, s_m$  be representatives of each equivalence class in  $S$ . On input  $\langle x, y \rangle$ , for each  $i \in \{1, \dots, n\}$  in parallel, determine if  $x \in [r_i]_R$  and  $y \in [r_i]_R$  (also in parallel). If  $x$  and  $y$  are both in  $[r_i]_R$  for some  $i$ , output  $\langle s_i, s_i \rangle$ , otherwise output  $\langle s_1, s_2 \rangle$ .

Since each string must be in exactly one of the equivalence classes of  $R$ , this function must halt when searching for the equivalence class for the strings  $x$  and  $y$ . If  $\langle x, y \rangle \in R$ , then they are in the same equivalence class of  $R$  and hence the function will output  $\langle s_i, s_i \rangle$ , which is in  $S$  by the reflexivity of  $S$ . If  $\langle x, y \rangle \notin R$ , then they are in different equivalence classes and hence the function will output  $\langle s_1, s_2 \rangle$ , which is not in  $S$  because  $[s_1]_S \neq [s_2]_S$  by hypothesis. Therefore this function is a computable many-one reduction from  $R$  to  $S$ .  $\square$

**Example 4.4.** Let  $R = \mathbb{Z}/3\mathbb{Z}$  and  $S = \mathbb{Z}/2\mathbb{Z}$ . Then  $R \leq_m S$  but  $R \not\leq_{ker} S$ .

As seen in Proposition 4.3, for equivalence relations  $R$  and  $S$  with a finite number of equivalence classes, a kernel reduction from  $R$  to  $S$  can only exist if the number of equivalence classes in  $R$  is at most the number of equivalence classes in  $S$ . However, most “interesting” equivalence relations have an infinite number of equivalence classes. In [12, Section 4], the authors ask if there are such equivalence relations “of the same densities [that is, density of equivalence classes] on which kernel reduction and [many-one] reduction differ.” [6, Theorem 5.1] (see also [6, Remark 5.2]) answers this question affirmatively, providing an infinite

antichain of equivalence relations that are equivalent under polynomial-time many-one reductions but otherwise incomparable under polynomial-time “strong isomorphism reductions” (proven in [6, Section 7] to be equivalent to polynomial-time kernel reductions). We will provide a simple proof of a special case of [6, Theorem 5.1], showing that an imbalance in the density of equivalence classes prevents a kernel reduction. This proof is valuable because it requires only knowledge of basic computational complexity theory and not knowledge of Boolean algebras, descriptive set theory, or other mathematical logic.

First we show that an equivalence relation dense in equivalence classes cannot be reduced to one sparse in equivalence classes. We emphasize that our result does not concern the sparseness of strings in a language, but the sparseness of equivalence classes in an equivalence relation. This complements the work on “potential reducibility” defined in [6, Section 5].

**Definition 4.5** ([6, Definition 7.2]). Let  $R$  and  $S$  be equivalence relations on  $\Sigma^*$ .  $R$  is *potentially reducible* to  $S$ , denoted  $R \leq_{pot}^P S$ , if there exists a polynomial  $p$  such that for all  $n \in \mathbb{N}$ ,  $\#R(n) \leq \#S(p(n))$ .

It follows from the definitions that for any equivalence relations  $R$  and  $S$ ,  $R \leq_{ker}^P S \implies R \leq_{pot}^P S$ , and hence  $R \not\leq_{pot}^P S \implies R \not\leq_{ker}^P S$  (this is stated and proven explicitly in [6, Lemma 5.5]). As an analog to traditional sparse languages, we provide a definition of “kernel sparsity,” and show its application to determining potential reducibility and hence kernel reducibility.

**Definition 4.6.** An equivalence relation  $R$  on  $\Sigma^*$  is *kernel sparse* if there exists a polynomial  $p$  such that for all  $n \in \mathbb{N}$ ,  $\#R(n) \leq p(n)$ . In other words, the number of equivalence classes in  $R$  for strings of length at most  $n$  is bounded above by a polynomial in  $n$ .

An equivalence relation is *kernel dense* if it is not kernel sparse. Formally, if for all polynomials  $p$  there exists an  $n \in \mathbb{N}$  such that  $\#R(n) > p(n)$ . In other words, the number of equivalence classes in  $R$  for strings of length at most  $n$  is greater than any polynomial in  $n$ .

These definitions allow us to provide the following very natural proposition. Intuitively, it states that an equivalence relation with many closely packed equivalence classes cannot reduce (under polynomially bounded notions of reduction) to an equivalence relation with few but widely spaced equivalence classes. This idea is stated without proof in [12, Section 4], so we provide it here for completeness. It is also essentially a special case of [14, Lemma 2.3], developed independently of that paper.

**Theorem 4.7.** Let  $R$  and  $S$  be equivalence relations on  $\Sigma^*$ . If  $R$  is kernel dense and  $S$  is kernel sparse, then  $R \not\leq_{ker}^P S$ .

*Proof.* That  $R \not\leq_{pot}^P S$  implies  $R \not\leq_{ker}^P S$  was already stated in the text preceding this theorem, so it suffices to show that  $R \not\leq_{pot}^P S$ .

Assume that  $R \leq_{pot}^P S$  with the intention of producing a contradiction. Let  $p$  be a polynomial such that  $\#R(n) \leq \#S(p(n))$  (this is the definition of potential

reducibility). Let  $q$  be a polynomial such that  $\#S(n) < q(n)$  for each natural number  $n$  (this is the definition of kernel sparse). Substituting  $p(n)$  for  $n$  in this inequality yields the inequality  $\#S(p(n)) < q(p(n))$ , which is a polynomial in  $n$ . Let  $r = q \circ p$ .

Let  $n_0$  be a natural number such that  $\#R(n_0) > r(n_0)$ , by the definition of kernel sparsity. Since  $\#S(p(n_0)) \leq r(n_0)$ , we have  $\#R(n_0) > \#S(p(n_0))$ . In other words, there are more equivalence classes in  $R$  for strings up to length  $n_0$  than there are in  $S$  for strings up to length  $p(n_0)$ . By the pigeonhole principle, we conclude that  $R$  cannot potentially reduce to  $S$ , because the number of equivalence classes in  $R$  for strings up to length  $n_0$  is too great compared to the number of equivalence classes in  $S$  for strings up to length  $p(n_0)$ . This is a contradiction with the assumption that  $R \leq_{pot}^P S$ . We have shown this for arbitrary polynomials (which came from the definitions of potential reducibility and kernel sparsity), so we can conclude that the result holds for all equivalence relations  $R$  and  $S$  that are kernel dense and kernel sparse, respectively.  $\square$

**Example 4.8.** Consider the equality relation and the “equal lengths” relation (that is,  $x$  relates to  $y$  if  $|x| = |y|$ ). The equality relation is kernel dense, since there are  $2^n$  equivalence classes for strings of length at most  $n$  (one for each string). The “equal lengths” relation is kernel sparse, since there are  $n + 1$  equivalence classes for strings of length at most  $n$  (one for each length, including length 0). Therefore there is no polynomial-time kernel reduction from the equality relation to the “equal lengths” relation.

This places a strong restriction on equivalence relations that are hard (or complete) under polynomial-time kernel reductions: they cannot be kernel sparse. This means that the equality relation, the densest possible equivalence relation with an exponential number of equivalence classes at each length, is a troublemaker in every complexity class that contains it.

**Corollary 4.9.** *Let  $\mathcal{CEq}$  be a complexity class of equivalence relations containing the equality relation  $R_{eq}$ . If an equivalence relation  $R$  is kernel sparse, then it is not  $\mathcal{CEq}$ -hard.*

Polynomial-time many-one reductions are more powerful than polynomial-time kernel reductions because the former are not subject to restrictions on numbers of equivalence classes as in Proposition 4.3 and Theorem 4.7. The idea behind Theorem 4.7 leads to a construction of equivalence relations  $R$  and  $S$  between which there is a polynomial-time many-one reduction but no polynomial-time kernel reduction.

**Construction 4.10.** Let  $f_1, f_2, \dots$  be an enumeration of all polynomial-time computable functions. Assume, without loss of generality, that for all positive integers  $i$ , function  $f_i$  runs in time  $p_i(n)$ , where  $p_i(n) = in^i$  for all positive integers  $n$ .

Suppose  $n$  is a positive integer. Define  $R_n$  as the set of all strings of length  $n$ , except  $R_1$ , which also includes the string of length 0. Define  $S_n$  as the set of

all strings  $s$  satisfying the inequality  $p_n(n) + 1 \leq |s| \leq p_{n+1}(n+1)$ , except  $S_1$ , which includes all strings of length at most  $p_2(2)$ .

Define sets  $R$  and  $S$  as

$$R = \bigcup_{n \in \mathbb{Z}^+} R_n \times R_n \text{ and } S = \bigcup_{n \in \mathbb{Z}^+} S_n \times S_n. \quad \square$$

**Lemma 4.11.**  *$R$  and  $S$  are equivalence relations.*

*Proof.*  $R$  and  $S$  are equivalence relations if  $\{R_n\}_{n \in \mathbb{Z}^+}$  and  $\{S_n\}_{n \in \mathbb{Z}^+}$  are valid partitions of  $\Sigma^*$ , so it suffices to show that the union of each collection includes all nonempty strings in  $\Sigma^*$  and that each collection is pairwise disjoint.

For  $\{R_n\}_n$ , any string of length  $n$  is in  $R_n$ , so  $\Sigma^* \subseteq \bigcup_n R_n$ . If  $m$  and  $n$  are distinct positive integers, no string can have both length  $m$  and length  $n$ , so  $R_m \cap R_n = \emptyset$ . Hence  $\{R_n\}_n$  is a valid partition.

For  $\{S_n\}_n$ , for any string  $x$ , there is an  $n$  such that  $p_n(n) + 1 \leq |x| \leq p_{n+1}(n+1)$ , so every string in  $\Sigma^*$  is in some  $S_n$ . To show pairwise disjointness, suppose  $m$  and  $n$  are distinct positive integers and assume without loss of generality that  $m < n$ , or in other words, that  $m+1 \leq n$ . Then  $p_{m+1}(m+1) \leq p_n(n) < p_n(n)+1$ , so no string of length at most  $p_{m+1}(m+1)$  can also have length at least  $p_n(n)+1$ . Hence,  $S_m$  and  $S_n$  are disjoint. Thus,  $\{S_n\}_n$  is a valid partition.

Since both collections are valid partitions, the relations  $R$  and  $S$  are both equivalence relations.  $\square$

Again, this is a special case of [6, Theorem 5.1], but has a much simpler proof and sufficiently demonstrates that polynomial-time kernel reductions and polynomial-time many-one reductions are different.

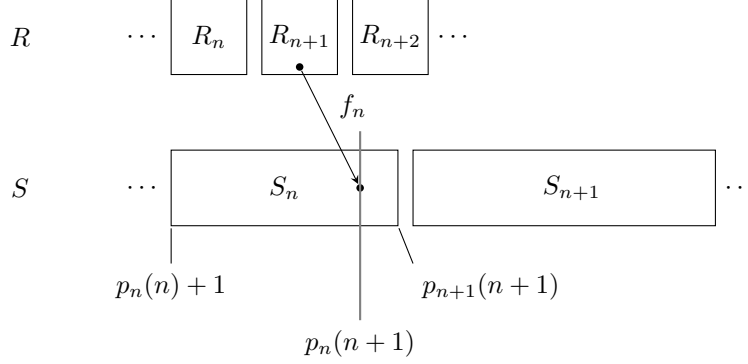
**Theorem 4.12.** *There are equivalence relations  $R$  and  $S$  such that  $R \leq_m^P S$  but  $R \not\leq_{ker}^P S$ . Furthermore,  $R$  and  $S$  are in  $\text{NC}^1\text{Eq}$ .*

The main idea behind this theorem is that no matter which polynomial-time function we consider as a possible kernel reduction, the number of equivalence classes in  $R$  is greater than the number of equivalence classes in  $S$ , for sufficiently large strings. Theorem 4.7 doesn't apply in this setting because both  $R$  and  $S$  are kernel sparse. Since we have carefully constructed these sets,  $S$  is more kernel sparse than  $R$ . This basic idea was presented independently in [14, Lemma 2.3].

Though it is not explicitly stated here, this theorem can be generalized to kernel reductions with other (non-polynomial) time bounds in a straightforward manner.

*Proof of Theorem 4.12.* Let  $R$  and  $S$  be the equivalence relations in Construction 4.10. The following function is a polynomial-time many-one reduction from  $R$  to  $S$ . On input  $\langle x, y \rangle$ , if  $|x| = |y|$  (or if  $|x|$  and  $|y|$  are both in  $\{0, 1\}$ ), output  $\langle a, a \rangle$ , otherwise output  $\langle a, b \rangle$ , where  $a$  is a string in  $S_1$  and  $b$  is a string in  $S_2$ . Computing and comparing the lengths of  $x$  and  $y$  can be done in linear time and writing the output requires only a constant number of steps, since the lengths of  $a$  and  $b$  are independent of the lengths of  $x$  and  $y$ . The correctness of the

Figure 1: For a fixed kernel reduction  $f_n$  running in time  $p_n$ , the image of a string of length  $n + 1$  can only be a string of length at most  $p_n(n + 1)$ . The number of equivalence classes in  $R$  for strings of length  $n + 1$  is greater than the number of equivalence classes in  $S$  for strings of length  $p_n(n + 1)$  for each polynomial  $p_n$ .



reduction follows from the fact that  $a$  and  $b$  are in different equivalence classes. Therefore there is a polynomial-time many-one reduction from  $R$  to  $S$ .

Now assume with the intention of producing a contradiction that there is a polynomial-time kernel reduction from  $R$  to  $S$ . Since  $f_1, f_2, \dots$  is an enumeration of all polynomial-time computable functions, the reduction from  $R$  to  $S$  is  $f_n$ , with running time  $p_n$ , for some positive integer  $n$ . Consider a string  $x$  of length  $n + 1$  (for example,  $x = 1^{n+1}$ );  $x$  is in equivalence class  $R_{n+1}$ . Since the running time of  $f_n$  is  $p_n$ , the length of  $f_n(x)$  is at most  $p_n(n + 1)$ . Since  $p_1, p_2, \dots$  is an increasing sequence (in the sense that  $p_j(n) < p_{j+1}(n)$  for all natural numbers  $n$  and all positive integers  $j$ ), we have  $p_n(n + 1) < p_{n+1}(n + 1) < p_{n+1}(n + 1) + 1$ . By the construction of  $R$  and  $S$ , we have  $\#R(n + 1) = n + 1$  and  $\#S(p_n(n + 1)) \leq \#S(p_{n+1}(n + 1)) = n$  (for an illustration, see Figure 1). By the pigeonhole principle, there must be two strings  $x$  and  $y$  of length at most  $n + 1$  in different equivalence classes of  $R$  whose image under  $f_n$  is in the same equivalence class of  $S$ . Since  $\langle x, y \rangle \notin R$  if and only if  $\langle f(x), f(y) \rangle \notin S$ , this is a contradiction. Therefore  $R \not\leq_{ker}^P S$ .

Finally, we show that  $R$  and  $S$  are in  $\text{NC}^1$ . Deciding whether two strings have the same length is trivial, so  $R$  is certainly in  $\text{NC}^1$ . To decide  $S$ , we compute the index  $i$  of the equivalence class  $S_i$  containing the string  $x$  and the index  $j$  of the equivalence class  $S_j$  containing the string  $y$ , then compare them for equality. Computing the index  $i$  of the equivalence class of a string  $x$  of length  $n$  can be performed as follows. First, compute in parallel the values  $p_1(1), \dots, p_{n+1}(n + 1)$ . Since each  $p_i$  is increasing,  $n$  is definitely smaller than  $p_{n+1}(n + 1)$ . Computing the exponentiation of  $O(n)$  pairs of strings of length  $O(n)$  each can be performed by a  $\text{TC}^0$  circuit, and  $\text{TC}^0 \subseteq \text{NC}^1$ . Next, for each  $i \in \{1, \dots, n\}$  in parallel,

decide if  $p_i(i) + 1 \leq n \leq p_{i+1}(i + 1)$ , thereby determining whether the input is in  $S_i$ . These comparisons can be performed by an  $\text{NC}^1$  circuit. Finally, use  $O(\log n)$  single-bit multiplexers in parallel to output the index (in binary) of the sole equivalence class  $S_i$  containing  $x$ . A single-bit multiplexer for  $O(\log n)$  input bits can be implemented by a circuit of size  $O(\log n)$  and depth  $O(\log \log n)$  [22, Lemma 2.5.5], so this phase of the computation can be performed by an  $\text{NC}^1$  circuit. Computing the indices  $i$  and  $j$  for the two input strings can be performed in parallel, and the final comparison for equality of  $i$  and  $j$  adds only  $O(\log n)$  depth to the circuit. Therefore  $S \in \text{NC}^1$ .  $\square$

## 5 Conditions for complete problems under polynomial-time kernel reductions

Most well-behaved complexity classes contain problems that are complete under many-one reductions. Do the corresponding classes of equivalence problems contain problems that are complete under kernel reductions? Having access to a complete problem offers many benefits and improves our understanding of equivalence problems in general. In [6, Theorem 8.7], the authors constructed a complete problem with respect to polynomial-time kernel reductions for  $\text{NPEq}$  under the assumption that  $\text{NP} = \text{coNP}$ . Since we consider that assumption unlikely, we determine sufficient conditions for having a complete problem under polynomial-time kernel reductions. This section presents a more general theorem that implies as a corollary a complete problem for  $\text{NPEq}$  under the assumption  $\text{NP} = \text{coNP}$ .

By extending the technique of [6, Theorem 8.7], we find that  $\text{PSPACEEq}$  has a complete problem under polynomial-time kernel reductions unconditionally. We also show that each level of the polynomial-time hierarchy contains an equivalence problem that is hard for the lower levels under these reductions. This means that some well-known classes do have complete problems, and the existence for complete problems in other classes, like  $\text{NP}$  and even  $\text{P}$ , remains possible. The existence of a natural complete problem remains open.

We need one additional definition in order to describe the complexity classes that contain a hard problem under kernel reductions. If  $\mathcal{C}$  is a complexity class then the class  $\forall\mathcal{C}$  is the set of languages  $A$  such that there exists a language  $B \in \mathcal{C}$  and a polynomial  $p$  satisfying  $x \in A$  if and only if  $\forall w \in \Sigma^{\leq p(|x|)} \langle x, w \rangle \in B$ .  $\forall\mathcal{C}$  is called the *closure of  $\mathcal{C}$  under polynomially bounded universal quantification*.

**Theorem 5.1.** *Let  $\mathcal{C}$  be a subset of  $\text{PSPACE}$  which contains the problem of deciding whether two strings are equal. Then there exists an equivalence relation in  $(\forall(\mathcal{C} \cup \text{co}\mathcal{C}))\text{Eq}$  which is hard for  $\mathcal{CEq}$  under  $\leq_{\text{ker}}^P$  reductions.*

Before proving this theorem, we will provide some immediate corollaries of this general result.

**Corollary 5.2.** *If  $\mathcal{C}$  is a subset of  $\text{PSPACE}$  and  $\mathcal{C} = \forall(\mathcal{C} \cup \text{co}\mathcal{C})$ , then  $\mathcal{CEq}$  has a complete problem under  $\leq_{\text{ker}}^P$  reductions.*

**Corollary 5.3.** *Under polynomial-time kernel reductions,*

1.  $\text{PSPACEEq}$  has a complete problem,
2.  $\Pi_k\text{PEq}$  contains a problem that is hard for  $\Delta_k\text{PEq}$ , for all  $k \geq 1$ .

*Proof.*

1.  $\text{PSPACE}$  is closed under complement (because it is a deterministic complexity class) and polynomially bounded universal quantification (because we can simulate the universal guess deterministically in polynomial space).
2. First,  $(\forall(\Delta_k P \cup \text{co}\Delta_k P))\text{Eq} = (\forall\Delta_k P)\text{Eq}$ , since  $\Delta_k P$  is closed under complement. Next,  $(\forall\Delta_k P)\text{Eq} = \Pi_k\text{PEq}$ , since  $\forall\Delta_k P = \Pi_k P$ . Now if we choose  $C = \Delta_k P$  in Theorem 5.1, then  $\Pi_k\text{PEq}$  has a problem that is hard for  $\Delta_k\text{PEq}$  under  $\leq_{ker}^P$  reductions.  $\square$

More specifically, this means that  $\text{coNPEq}$  (which equals  $\Pi_1\text{PEq}$ ) has a problem that is  $\leq_{ker}^P$ -hard for  $\text{PEq}$  (which equals  $\Delta_1\text{PEq}$ ). This corollary also leads to [6, Theorem 8.7, part 1], which is restated here.

**Corollary 5.4** ([6, Theorem 8.7, part 1]). *If  $\text{NP} = \text{coNP}$  then  $\text{NPEq}$  has a complete problem under polynomial-time kernel reductions.*

*Proof.* If  $\text{NP} = \text{coNP}$ , then the polynomial hierarchy collapses to  $\Pi_1 P$ , and specifically  $\Pi_2 P = \Delta_2 P = \Pi_1 P = \text{coNP} = \text{NP}$ . From Corollary 5.3 we conclude that  $\text{NPEq}$  has a  $\leq_{ker}^P$ -hard problem for  $\text{NPEq}$ . Such a problem is by definition  $\text{NPEq}$ -complete.  $\square$

We now return to the proof of Theorem 5.1 by first providing some motivating ideas. Recall the canonical complete problem (sometimes called the “universal” problem) for  $\text{NP}$  (and indeed for various other complexity classes):

$$K = \{ \langle M, x, 1^t \rangle \mid M \text{ accepts } x \text{ within } t \text{ steps} \}$$

The idea of this proof is to adapt this into an equivalence relation  $R_K$  consisting of pairs of triples of the form  $\langle \langle M, x, 1^{t_x} \rangle, \langle M, y, 1^{t_y} \rangle \rangle$ , where  $M$  accepts  $\langle x, y \rangle$ , as in the reduction from an arbitrary  $\text{NP}$  language to  $K$ . The problem we encounter here is that  $R_K$  is not necessarily an equivalence relation. Consider, for example, transitivity, which must be satisfied for all possible pairs of the form  $\langle M, w, 1^{t_w} \rangle$ . For *arbitrary machines*  $M$ , just because  $M$  accepts  $\langle x, y \rangle$  and  $\langle y, z \rangle$  does not necessarily mean that  $M$  accepts  $\langle x, z \rangle$ . The solution is to encode into  $R_K$  the requirement that the language which  $M$  accepts,  $L(M)$ , is itself an equivalence relation. The three properties required of  $R_K$  then follow from the properties of  $L(M)$ .

*Proof of Theorem 5.1.* First we will define a helper algorithm which decides whether a given machine accepts an equivalence relation on strings up to a given length. Define the algorithm  $A$  as follows on input  $\langle M, n \rangle$ , where  $M$  is a polynomially clocked Turing machine of type  $C$  and  $n \in \mathbb{N}$ :

1. universally guess  $a, b$ , and  $c \in \Sigma^{\leq n}$ ,
2. simulate  $M$  on  $\langle a, a \rangle$ ; if it rejects, reject,
3. simulate  $M$  on  $\langle a, b \rangle$ , then on  $\langle b, a \rangle$ ; if the former accepts and the latter rejects, reject,
4. simulate  $M$  on  $\langle a, b \rangle$ , then on  $\langle b, c \rangle$ , then on  $\langle a, c \rangle$ ; if the first two accept and the last one rejects, reject,
5. if execution reaches this point, accept.

These simulations check that  $L(M)$  satisfies reflexivity, symmetry, and transitivity on strings of length at most  $n$ . If  $A$  accepts, then the three properties are satisfied, and if it rejects then one of the three properties is violated. Since  $M$  is a machine of type  $\mathcal{C}$ , checking if  $M$  accepts on some input and if  $M$  rejects on some input is in  $\mathcal{C} \cup \text{co}\mathcal{C}$ . The universal guesses of  $a, b$ , and  $c$  (of length at most  $n$ ) followed by checks of whether the six simulations of  $M$  accept or reject place  $L(A)$  in the class  $\forall(\mathcal{C} \cup \text{co}\mathcal{C})$ . If  $p$  is the polynomial which bounds the running time of  $M$ , then the running time of this algorithm is  $6p(|\langle 1^n, 1^n \rangle|) + c$ , where  $c$  is a constant which represents the time needed to account for the implementation of  $A$  (the control of the simulations of  $M$ , performing logical conjunctions, etc.). Hence the running time of  $A$  is polynomial in  $n$ .

Now we can define the set  $R_K$  as follows. A pair of strings  $\langle u, v \rangle$  is in  $R_K$  if and only if either  $u = v$  or  $u$  and  $v$ , when interpreted as strings of the form  $\langle M, x, 1^{t_x} \rangle$  and  $\langle M, y, 1^{t_y} \rangle$ , respectively, satisfy the four conditions

1.  $M$  is a polynomially clocked Turing machine of type  $\mathcal{C}$ ,
2.  $A$  accepts  $\langle M, |x| \rangle$  within  $t_x$  steps,
3.  $A$  accepts  $\langle M, |y| \rangle$  within  $t_y$  steps,
4.  $M$  accepts  $\langle x, y \rangle$ .

We claim that  $R_K$  is in  $(\forall(\mathcal{C} \cup \text{co}\mathcal{C}))\text{Eq}$  and  $\mathcal{C}\text{Eq-hard}$ .

First we show that  $R_K \in \forall(\mathcal{C} \cup \text{co}\mathcal{C})$ . By the argument above,  $A$  is a  $\forall(\mathcal{C} \cup \text{co}\mathcal{C})$  algorithm. Assuming without loss of generality that  $|x| \geq |y|$ , if  $A$  accepts  $\langle M, |x| \rangle$  within  $t_x$  steps then we know that there is a polynomial-time bound on the running time of  $M$  on input  $\langle x, y \rangle$ , so simulating it is certainly in  $\forall(\mathcal{C} \cup \text{co}\mathcal{C})$ . Finally, testing for equality is in  $\mathcal{C}$  by hypothesis so deciding  $R_K$  overall can be performed by a  $\forall(\mathcal{C} \cup \text{co}\mathcal{C})$  algorithm.

Next we show that  $R_K$  is an equivalence relation. Reflexivity follows from the reflexivity of the equality relation. For symmetry, suppose that the pair  $\langle \langle M, x, 1^{t_x} \rangle, \langle M, y, 1^{t_y} \rangle \rangle$  is in  $R_K$ . Since item 2 and item 3 are true by hypothesis, we know that symmetry on strings of length at most  $\max(|x|, |y|)$  in  $L(M)$  is satisfied, and that includes the strings  $x$  and  $y$ . So since  $M$  accepts  $\langle x, y \rangle$  it must follow that  $M$  accepts  $\langle y, x \rangle$ . Furthermore, item 1, item 2, and item 3 are the same up to symmetry of  $x$  and  $y$ , so we have  $\langle \langle M, y, 1^{t_y} \rangle, \langle M, x, 1^{t_x} \rangle \rangle \in$



$R_K$ . For transitivity, suppose that both  $\langle \langle M, x, 1^{t_x} \rangle, \langle M, y, 1^{t_y} \rangle \rangle \in R_K$  and  $\langle \langle M, y, 1^{t_y} \rangle, \langle M, z, 1^{t_z} \rangle \rangle \in R_K$ . Since transitivity is true on strings of length at most  $\max(|x|, |y|, |z|)$  by the transitivity propositions checked by item 2 and item 3, and since  $M$  accepts both  $\langle x, y \rangle$  and  $\langle y, z \rangle$  by hypothesis, it must follow that  $M$  accepts  $\langle x, z \rangle$ . Again the conditions in item 1, item 2, and item 3 are the same. We have shown that  $R_K$  is reflexive, symmetric, and transitive, so it is an equivalence relation. At this point, we have proven that  $R_K \in (\forall(\mathcal{C} \cup \text{co}\mathcal{C}))\text{Eq}$ .

Now we need to show that  $R_K$  is  $\mathcal{CEq}$ -hard. Let  $S \in \mathcal{CEq}$ . Suppose  $M$  is the polynomially clocked  $\mathcal{C}$  machine that decides  $S$ , and  $p$  is the polynomial that bounds the running time of  $M$ . Then the kernel reduction from  $S$  to  $R_K$  is  $w \mapsto \langle M, w, 1^{6p(|\langle w, w \rangle|)+c} \rangle$ , where  $p$  and  $c$  are the polynomial and constant described in the first paragraph of this proof. Call this reduction  $f$ . The reduction is obviously computable in time polynomial in  $|w|$ . It remains to show that this reduction is correct.

Suppose  $\langle x, y \rangle \in S$ . Now  $f(x) = \langle M, x, 1^{6p(|\langle x, x \rangle|)+c} \rangle$  and, similarly,  $f(y) = \langle M, y, 1^{6p(|\langle y, y \rangle|)+c} \rangle$ . item 1 is true by construction, and item 4 is true since  $M$  is the machine which decides  $S$ . Assume item 2 is false. Then  $M$  does not accept an equivalence relation on strings of length at most  $|x|$ . This is a contradiction, since  $M$  decides  $S$ , an equivalence relation, by hypothesis. Therefore item 2 must be satisfied. The same argument applies to item 3. Hence  $\langle f(x), f(y) \rangle \in R_K$ .

If  $\langle x, y \rangle \notin S$  then  $M$  does not accept  $\langle x, y \rangle$ , since otherwise  $\langle x, y \rangle$  would be a member of  $S$ . Hence  $\langle x, y \rangle \notin R_K$ . Therefore we have shown that  $R_K$  is  $\mathcal{CEq}$ -hard.  $\square$

**Open problem 5.5.** Is there a more general characterization of complexity classes which have a  $\leq_{ker}^P$ -hard problem?

**Open problem 5.6.** Is there an oracle relative to which  $\text{PEq}$  or  $\text{NPEq}$  has a complete problem under polynomial-time kernel reductions? We conjecture that  $\text{NPEq}$  has a complete problem without relativization.

**Open problem 5.7.** Is the converse of Corollary 5.2, or perhaps a partial converse, true? In other words, is it true that the existence of a  $\mathcal{CEq}$ -complete problem implies closure under complement or universal quantification (or both)? If so, this would be evidence that no  $\text{NPEq}$ -complete problem exists, since this would imply  $\text{NP} = \text{coNP}$ .

**Open problem 5.8.** Can this theorem be used to construct  $\leq_{ker}$ -hard problems for smaller complexity classes such as  $\text{NLEq}$  under the appropriate time-bounded reduction? Larger classes such as  $\text{EXPEq}$ ?

**Open problem 5.9.** To what other equivalence relations does our  $\leq_{ker}^P$ -hard problem reduce? Are there “natural”  $\leq_{ker}^P$ -hard problems in complexity classes which satisfy the conditions in Theorem 5.1?

## 6 Relationship between completeness under kernel and many-one reductions

A kernel reduction implies a many-one reduction, but does completeness under kernel reductions imply completeness under many-one reductions? Since polynomial-time kernel reductions are different from polynomial-time many-one reductions (Theorem 4.12), completeness in classes of equivalence problems may differ under these reductions as well. We determine the conditions under which completeness under kernel reductions implies completeness under many-one reductions.

We find that completeness under many-one reductions follows as a straightforward consequence of completeness under kernel reductions as long as the relevant complexity class admits a complete problem under many-one reductions. We also show that the kernel reduction is essentially too weak to allow for completeness under injective (that is, “one-to-one”) reductions, for combinatorial reasons similar to those in section 4. Though we prove these results for **NPEq**, they generalize in a natural way to any “well-behaved” complexity class (basically, any class containing a complete problem under many-one reductions). These results are more indication that when comparing the relative difficulty of equivalence problems, one should attempt to construct a kernel reduction instead of a many-one reduction. The potential lack of a complete problem under injective kernel reductions suggests that a conjecture analogous to the Berman–Hartmanis conjecture, which states that all **NP**-complete problems are isomorphic with respect to many-one reductions, may be false in **NPEq**.

One can infer the existence of **NP**-complete equivalence relations from the relation suggested in [12, Section 6.2],

$$\{\langle 0\phi, 1\phi \rangle \mid \phi \in \text{SATISFIABILITY}\}.$$

(This relation is not itself an equivalence relation, but can be modified to guarantee the three necessary properties.) Using this idea, we provide a strategy for constructing a more natural **NP**-complete equivalence relation from an equivalence relation in **NP** and an arbitrary **NP**-complete property.

Let **GI** denote the equivalence relation consisting of all pairs of isomorphic graphs. A property, that is, a Boolean function,  $\Pi$  is an *NP-complete property* if  $L_\Pi$ , the set of all strings for which  $\Pi$  is true, is **NP**-complete. If, furthermore, the property satisfies  $\langle x, y \rangle \in R$  implies  $\Pi(x) = \Pi(y)$  where  $R$  is an equivalence relation,  $\Pi$  is called a *property on R*. For example, Hamiltonicity, the property of having a cycle that includes each vertex, is an **NP**-complete property on **GI**.

**Theorem 6.1.** *If  $\Pi$  is an **NP**-complete property on **GI**, then the equivalence relation  $A$  defined by*

$$A = \{\langle G, H \rangle \mid \langle G, H \rangle \in \text{GI or } (G \in L_\Pi \text{ and } H \in L_\Pi)\}$$

*is an **NP**-complete equivalence relation.*

*Proof.* It is straightforward to prove that  $A$  is an equivalence relation, so it remains to show that it is **NP**-complete. The language  $A$  is in **NP** because both  $R$  and  $L_\Pi$  are in **NP** by hypothesis. Thus we need only show that  $A$  is **NP**-hard.

Let  $H$  be a graph satisfying  $\Pi$ ; such a graph must exist because  $\Pi$  is **NP**-complete and therefore there must be at least one graph that satisfies  $\Pi$  and at least one that does not (otherwise no many-one reduction to  $L_\Pi$  could exist). The reduction proving that  $A$  is **NP**-complete is from  $L_\Pi$ , and the mapping is given by  $G \mapsto \langle G, H \rangle$ . This function is computable in linear time; the size of  $H$  is constant with respect to the size of  $G$ .

Now we show that  $G \in L_\Pi$  if and only if  $\langle G, H \rangle \in A$ , for any graph  $G$ . If  $G \in L_\Pi$ , then  $G \in L_\Pi$  and  $H \in L_\Pi$ , so  $\langle G, H \rangle \in A$ . If  $\langle G, H \rangle \in A$ , then either  $G \in L_\Pi$  and  $H \in L_\Pi$ , in which case  $G \in L_\Pi$ , or  $G$  is isomorphic to  $H$ , in which case  $G$  is in  $L_\Pi$  because  $H$  is. In either case  $G \in L_\Pi$ . We conclude that  $L_\Pi \leq_m^P A$ , and so  $A$  is an **NP**-complete equivalence relation.  $\square$

**Example 6.2.** The language

$$\{\langle G, H \rangle \mid \langle G, H \rangle \in \text{GI or } G \text{ and } H \text{ have a Hamiltonian cycle}\}$$

is an **NP**-complete equivalence relation.

There are other ways of constructing a natural **NP**-complete equivalence relation. For example, there is a finitely presented group whose word problem is **NP**-complete [21, Corollary 1.1], and the word problem is already an equivalence relation. This may be considered “more natural” because it does not involve the disjunction of two distinct computational problems, though it lacks the simplicity of our approach.

**Example 6.3.** As stated briefly above, Theorem 6.1 can be generalized to isomorphism of structures other than graphs and/or larger complexity classes. For example, replacing GI with FI, the Boolean formula isomorphism problem, and an **NP**-complete property on GI with a  $\Sigma_2\text{P}$ -complete property on FI yields a  $\Sigma_2\text{P}$ -complete equivalence relation.

**Corollary 6.4.** *If  $R$  is **NPEq**-complete, then  $R$  is **NP**-complete.*

*Proof.* This follows immediately from the existence of an **NP**-complete equivalence relation, as in Example 6.2, and the fact that a kernel reduction implies a many-one reduction.  $\square$

This corollary provides a clearer proof of [6, Proposition 8.1].

**Corollary 6.5** ([6, Proposition 8.1]). *If GI is **NPEq**-complete then the polynomial hierarchy collapses to the second level, that is,  $\text{PH} = \Sigma_2\text{P} \cap \Pi_2\text{P}$ .*

*Proof.* By the previous corollary, if GI is **NPEq**-complete, then it is **NP**-complete, which implies the stated collapse (see [23]).  $\square$

Theorem 6.1 also provides a simple method for proving the equivalence of  $\text{P} = \text{NP}$  and  $\text{PEq} = \text{NPEq}$ .

**Theorem 6.6.**  $P = NP$  if and only if  $PEq = NPEq$ .

*Proof.* If  $P = NP$ , then  $PEq = NPEq$  by their definitions. Suppose now that  $PEq = NPEq$ . Let  $A$  denote the NP-complete equivalence relation defined in Example 6.2. Since  $A \in NPEq$  and  $PEq = NPEq$  by hypothesis,  $A \in PEq$ , and hence  $A \in P$ . Since  $P$  is closed under  $\leq_m^P$  reductions, any NP-complete problem in  $P$  implies  $P = NP$ .  $\square$

As stated in Open problem 5.6, we do not know whether an NPEq-complete problem exists. In the following theorem we describe an equivalence relation that, if it were NPEq-complete, would prove that injective kernel reductions are strictly weaker than general kernel reductions. This is interesting because it again demonstrates that the number and size of equivalence classes is important when considering the (im)possibility of polynomial-time kernel reductions between equivalence relations. In the following theorem, if an equivalence relation is “complete under  $\leq_{ker,1}^P$  reductions in NPEq” we mean that every equivalence relation in NPEq reduces to it by a polynomial-time computable kernel reduction which is also injective (that is, “one-to-one”).

**Theorem 6.7.** Let  $\Pi$  be a property on GI. If the equivalence relation  $A$  defined by

$$A = \{\langle G, H \rangle \mid \langle G, H \rangle \in GI \text{ or } (G \in L_\Pi \text{ and } H \in L_\Pi \text{ and } |G| = |H|)\}$$

is complete for NPEq under  $\leq_{ker}^P$  reductions, then  $A$  is not complete under  $\leq_{ker,1}^P$  reductions.

The only difference between the equivalence relation  $A$  defined here and the one defined in Theorem 6.1 is the requirement that  $|G| = |H|$ . This means that although the number of equivalence classes in  $A$  is infinite (at least one for each size), each of those equivalence classes is itself finite. In contrast, consider the equivalence relation  $S$  defined by

$$S = \{\langle x, y \rangle \mid x \text{ and } y \text{ have the same number of 1s}\}.$$

The equivalence relation  $S$  has an infinite number of equivalence classes:  $[1]$ ,  $[11]$ ,  $[111]$ , etc. Each equivalence class is itself infinite as well: for each  $w \in \Sigma^*$ , the equivalence class  $[w]$  contains  $w$ ,  $0w$ ,  $00w$ , etc.

*Proof of Theorem 6.7.* Let  $S$  be the equivalence relation defined in the preceding paragraph. The language  $S$  is decidable in linear time by a deterministic Turing machine, hence it is in NP. Since  $A$  is NPEq-complete by hypothesis,  $S \leq_{ker}^P A$ . Thus there is a polynomial-time computable function  $f$  such that  $\langle x, y \rangle \in S$  if and only if  $\langle f(x), f(y) \rangle \in A$ .

By the discussion preceding this theorem,  $[w]_S$  is infinite and  $[f(w)]_A$  is finite. By Lemma 4.2,  $f([w]_S) \subseteq [f(w)]_A$ . Consider  $f|_{[w]_S}$ , that is,  $f$  restricted to the domain  $[w]_S$ . Then  $f|_{[w]_S}$  is a mapping from the infinite set  $[w]_S$  to the finite set  $[f(w)]_A$ . By the pigeonhole principle,  $f|_{[w]_S}$  is not injective. Hence the unrestricted reduction  $f$  is not injective, and therefore  $A$  is not  $\leq_{ker,1}^P$ -complete in NPEq.  $\square$

## 7 Existence of intermediary problems

According to the seminal theorem by Ladner [19], if  $P \neq NP$ , then there are problems of intermediate complexity, in the sense that these problems are neither in  $P$  nor  $NP$ -complete. The theorem does not immediately imply a similar result for equivalence problems, since  $PEq$  is different from  $P$  and  $NPEq$  is different from  $NP$  (specifically, in each case, the latter contains problems that are not equivalence problems). Do kernel reductions induce the same rich structure between  $PEq$  and  $NPEq$  as do many-one reductions between  $P$  and  $NP$ ? We adapt a proof of Ladner's theorem from [9] (which has been attributed to Russell Impagliazzo) to classes of equivalence problems; this section details that adaptation.

The main theorem of this section is the existence of  $NPEq$ -intermediary problems under the assumption that  $PEq \neq NPEq$  (which is equivalent to the assumption  $P \neq NP$  by Theorem 6.6). We conclude that even though kernel reductions are strictly weaker than many-one reductions, they still preserve the hierarchies of problems of various computational complexities we expect from our understanding of traditional complexity classes. The graph isomorphism problem, as one of the few candidates for an  $NP$ -intermediary problem, may be the best candidate for a natural  $NPEq$ -intermediary problem as well.

This proof of Ladner's theorem for equivalence relations is a delayed diagonalization via progressive padding. First we define the equivalence relation performing the diagonalization and the corresponding padding function, then we show that this problem is neither in  $PEq$  nor  $NPEq$ -complete.

**Construction 7.1.** Let  $K$  be an  $NP$ -complete equivalence relation. We know that such equivalence relations exist by Theorem 6.1. Define the equivalence relation  $R$  by

$$R = \left\{ \left\langle x01^{p(n)-n-1}, y01^{p(n)-n-1} \right\rangle \mid \langle x, y \rangle \in K \text{ and } |x| = |y| = n \right\},$$

where  $p$  is a padding function that will be defined below. The equivalence relation  $R$  is a padded version of  $K$ .

Our goal is to define the function  $p$  so that  $R$  is not too hard and not too easy: its output should be large enough that  $R$  is not  $NPEq$ -complete but not so large that  $R$  is in  $P$ . For this we need an enumeration of each polynomially clocked Turing machine,  $\{M_i\}_i$ , where machine  $M_i$  halts within time  $in^i$  on inputs of length  $n$ . For any pair of strings  $x$  and  $y$ , we say a Turing machine  $M$  *disagrees with  $R$  on  $\langle x, y \rangle$*  if

- $M(\langle x, y \rangle)$  accepts and  $\langle x, y \rangle \notin R$ , or
- $M(\langle x, y \rangle)$  rejects and  $\langle x, y \rangle \in R$ .

We define  $p$  for each positive integer  $n$  by the following iterative process (and thus we implicitly define  $R$  iteratively as well). Initially, let  $i = 1$ , then perform the following steps for each  $n$  in order.

- Define  $p(n)$  to be  $n^i$ .
- Check if there is any pair of strings  $x$  and  $y$ , each of length at most  $\log \log n$ , such that  $M_i$  disagrees with  $R$  on  $\langle x, y \rangle$ . If any such pair exists and  $\lfloor \log \log n \rfloor$  is an integer not already seen, then increment  $i$ .  $\square$

The following three lemmas prove that this problem is of intermediate complexity if  $\text{PEq} \neq \text{NPEq}$ .

**Lemma 7.2.** *The function  $p$  in Construction 7.1 is computable in time polynomial in  $n$ .*

*Proof.* Computing  $p(n)$  requires computing  $p(1), p(2), \dots, p(n-1)$ ; if each of these  $n-1$  computations takes a polynomial amount of time, the total time required to compute  $f(n)$  remains polynomial in  $n$ , by induction. Since the strings  $x$  and  $y$  are of length at most  $\log \log n$ , the total number of iterations required to test all pairs of strings is polynomial in  $n$ . The simulation of  $M_i$  is computable in time  $i(\log \log n)^i$ , but  $i$  is at most  $\log \log n$ , since  $i$  can only be incremented at most  $\log \log n$  times. Using the fact that a polynomial in  $\log \log n$  is bounded above by  $O(\log n)$ ,

$$\begin{aligned}
i(\log \log n)^i &\leq \log \log n (\log \log n)^{\log \log n} \\
&= 2^{(\log \log \log n)^2 \log \log n} \\
&\leq 2^{(\log \log n)^2} \\
&= 2^{O(\log n)} \\
&= \text{poly}(n),
\end{aligned}$$

so the machine  $M_i$  runs in time polynomial in  $n$ . The language  $R$  is in  $\text{NPEq}$  because it is a padded version of the language  $K$ , which is in  $\text{NPEq}$ . Since  $\text{NP} \subseteq \text{EXP}$  and the inputs  $x$  and  $y$  are each of length  $\log \log n$ , membership in  $R$  can be determined in time polynomial in  $n$ . (Even though the definition of  $R$  requires  $p$  to be defined,  $p$  is already defined for strings of length less than  $n$ , including the strings  $x$  and  $y$ .) Since each step can be performed in polynomial time and there are at most  $n$  iterations required when defining  $p(n)$ , we conclude that  $p$  is computable in time polynomial in  $n$ .  $\square$

**Lemma 7.3.** *Suppose  $R$  is the equivalence relation in Construction 7.1. If  $\text{PEq} \neq \text{NPEq}$ , then  $R \notin \text{PEq}$ .*

*Proof.* Assume with the intention of producing a contradiction that  $R \in \text{PEq}$ . Thus there is a natural number  $i$  such that  $M_i$  decides  $R$ . For sufficiently large  $n$ , the machine  $M_i$  never disagrees with  $R$ , so  $p(n) = n^i$  for all sufficiently large  $n$ . Assuming without loss of generality that the string  $x$  and  $y$  are each of length  $n$ , this yields a polynomial-time kernel reduction from  $K$  to  $R$  via the function  $\langle x, y \rangle \mapsto \langle x01^{p(n)-n-1}, y01^{p(n)-n-1} \rangle$ . This mapping is polynomial-time computable because  $p(n) = n^i$  for all sufficiently large  $n$ , and  $i$  does not depend on  $n$ . Since  $\text{PEq}$  is closed under polynomial-time kernel reductions,  $K$  is in  $\text{PEq}$ ,

and hence  $\text{PEq} = \text{NPEq}$ , since  $K$  is  $\text{NPEq}$ -complete. This is a contradiction with the assumption that  $\text{PEq} \neq \text{NPEq}$ , hence  $R \notin \text{PEq}$ .  $\square$

**Lemma 7.4.** *Suppose  $R$  is the equivalence relation in Construction 7.1. If  $\text{PEq} \neq \text{NPEq}$ , then  $R$  is not  $\text{NPEq}$ -complete.*

*Proof.* Assume with the intention of producing a contradiction that  $R$  is  $\text{NPEq}$ -complete. Thus  $K \leq_{ker}^P R$ , so there is a function  $f$  such that  $f$  halts within  $n^j$  steps and for each string  $x$  and  $y$ , we have  $\langle x, y \rangle \in K$  if and only if  $\langle f(x), f(y) \rangle \in R$ . If the image of  $f$  were finite, then  $R$  would have a constant number of equivalence classes. In this case,  $R$  would be in  $\text{PEq}$ , and since  $\text{PEq}$  is closed under polynomial-time kernel reductions,  $K$  would be in  $\text{PEq}$  as well, a contradiction with the hypothesis that  $\text{PEq} \neq \text{NPEq}$ .

Suppose the image of  $f$  is infinite. We know  $f(w)$  must be of the form  $w01^{p(|w|)-n-1}$  for each string  $w$  of length  $n$ , so the length of  $f(w)$  is  $p(|w|)$ . There is a natural number  $n_0$  such that for each  $n \geq n_0$ , there is a positive integer  $k$  such that  $k$  is greater than  $j$  and for each string  $w$  of length  $n$ , we have  $|f(w)| = p(n) = n^k$ . (The integer  $k$  is strictly greater than  $j$ , since if it were less than or equal to  $j$ , the image of  $f$  would be finite.) Now we can construct a polynomial-time algorithm for  $K$ . Assume without loss of generality that all inputs are pairs of strings of equal length. On inputs of the form  $\langle x, y \rangle$ , proceed as follows.

- If  $|x| < n_0$  (or equivalently  $|y| < n_0$ ), decide whether  $\langle x, y \rangle \in K$  by examining a hardcoded lookup table for strings of length less than  $n_0$ .
- Compute  $f(x)$  and  $f(y)$ .
- If either  $|f(x)|$  or  $|f(y)|$  is not in the range of  $p$ , reject.
- Suppose  $f(x) = x'01^{p(m)-m-1}$  and  $f(y) = y'01^{p(m)-m-1}$ , where  $|x'| = |y'| = m$ . Invoke this algorithm recursively on input  $\langle x', y' \rangle$ .

Assuming for now that  $x'$  and  $y'$  are shorter than  $x$  and  $y$ . Then the correctness of this algorithm follows from the fact that

$$\langle x, y \rangle \in K \iff \langle f(x), f(y) \rangle \in R \iff \langle x', y' \rangle \in K.$$

Since the length of the inputs to the algorithm decrease on each recursive invocation, there are at most  $n$  recursive calls on inputs of pairs of strings of length  $n$ . Eventually the solution can be found in the hardcoded lookup table (the base case of the recursion). Each recursive invocation of the algorithm other than the base case requires computing  $f$  on an input of length  $n$  (twice), which can be done in polynomial time. Thus the overall time required for this algorithm is polynomial in  $n$ . This proves that  $K \in \text{P}$  and thus  $\text{P} = \text{NP}$ . Since  $\text{P} = \text{NP}$  if and only if  $\text{PEq} = \text{NPEq}$ , we have a contradiction.

Finally, we prove that  $|x'| < |x|$  (the proof that  $|y'| < |y|$  is the same), which we postponed from the previous paragraph. Due to its time bound,  $|f(x)| \leq |x|^j$

for any string  $x$ . By assumption,  $p(|x'|) = |x'|^k$ . By construction,  $p(|x'|) = |f(x)|$ . Combining these three relations yields the inequality

$$|x'|^k = p(|x'|) = |f(x)| \leq |x|^j,$$

so  $|x'| \leq |x|^{j/k} < |x|$ , since  $k > j$  and lengths must be natural numbers.  $\square$

Combining the preceding three lemmas yields Ladner's theorem for classes of equivalence relations.

**Theorem 7.5.** *If  $\text{PEq} \neq \text{NPEq}$ , then there is an equivalence relation in  $\text{NPEq}$  that is neither in  $\text{PEq}$  nor  $\text{NPEq}$ -complete.*

This technique can be generalized to other classes of equivalence relations, as long as the underlying machines for the smaller class can be enumerated and the larger class has an equivalence relation that is complete under many-one reductions. For example, we can produce equivalence relations between the polynomial hierarchy and  $\text{PSPACE}$ .

**Corollary 7.6.** *If  $\text{PHEq} \neq \text{PSPACEEq}$ , then there is an equivalence relation in  $\text{PSPACEEq}$  that is neither in  $\text{PHEq}$  nor  $\text{PSPACEEq}$ -complete.*

## 8 Conclusion

Throughout this work we have proven that kernel reductions are similar to many-one reductions in the most basic ways, but differ in some key aspects. Like many-one reductions, kernel reductions are transitive and have good closure properties. The class of equivalence problems in  $\text{PSPACE}$  has a complete problem under kernel reductions (Corollary 5.3). The equivalence of the two equalities  $\text{P} = \text{NP}$  and  $\text{PEq} = \text{NPEq}$  (Theorem 6.6) uses the similarity between many-one and kernel reductions. Just as many-one reductions allow the existence of  $\text{NP}$ -intermediary problems, kernel reductions allow for the possibility of  $\text{NPEq}$ -intermediary problems (Theorem 7.5). On the other hand, there are equivalence relations between which there is a many-one reduction but no kernel reduction (Theorem 4.12). Specifically, if there are more equivalence classes, up to strings of certain lengths, in  $R$  than in  $S$ , then no kernel reduction can exist. Finally, under some assumptions, there is an equivalence problem that is not complete for  $\text{NPEq}$  under injective kernel reductions (Theorem 6.7), whereas nearly every known  $\text{NP}$ -complete problem is isomorphic (the Berman–Hartmanis conjecture [4] states that *every*  $\text{NP}$ -complete problem is isomorphic).

The techniques used in this paper to show that kernel reductions are weaker than many-one reductions are combinatorial techniques (for example, comparing the numbers of equivalence classes). Combining these with other complexity theoretic and algebraic techniques has already proven useful: there is no polynomial-time kernel reduction from the graph isomorphism problem to the isomorphism problem for strongly regular graphs [2, Theorem 22]. This is interesting because even though the latter appears to be a difficult problem, no



polynomial-time many-one reduction from the former to the latter is expected to exist [2], and the lack of a kernel reduction is evidence in that direction.

Besides the open problems listed in section 5, we consider the following questions to be worth exploring.

- There are several problems of *inequivalence* in [15] listed as NP-complete or as PSPACE-complete. What do these problems have to do with NPEq-completeness, coNPEq-completeness, and PSPACEEq-completeness?
- When can results like [7, Theorem 1], for example, which shows that an equivalence relation is complete for  $L$  under many-one reductions via a reduction from a problem that is *not* an equivalence relation, be translated to a proof that the problem is complete under kernel reductions for the corresponding class of equivalence problems?
- In the case of the graph isomorphism problem, GI, the number  $\#GI(n)$ , in the notation of section 4, is the number of (pairwise) non-isomorphic graphs on at most  $n$  vertices. This differs from the conventional notation  $\#GI$  denoting the problem of counting the number of graphs isomorphic to a given graph. In other words, our notation counts the *number* of equivalence classes, whereas the latter counts the *size* of an equivalence class. For the graph isomorphism problem, computing the size of an equivalence class is Turing-equivalent to deciding whether two graphs are isomorphic [18, Theorem 1.24]. When is the problem of computing the size of an equivalence class Turing-equivalent to the problem of deciding equivalence?

## 9 Acknowledgments

The authors acknowledge the invaluable help provided by Josh Grochow and Steve Homer. We thank the anonymous reviewers of an earlier version of this paper for numerous corrections and stylistic suggestions. Specifically, we thank an anonymous reviewer for showing how to prove  $Cl_1 \subseteq NKer_1$ , and another anonymous reviewer for simplifying the proof of Theorem 4.7.

## References

- [1] Uri Andrews et al. “Universal computably enumerable equivalence relations”. In: *The Journal of Symbolic Logic* 79.01 (2014), pp. 60–88. DOI: 10.1017/jsl.2013.8.
- [2] László Babai. “On the Automorphism Groups of Strongly Regular Graphs I”. In: *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*. ITCs ’14. New York, NY, USA: ACM, 2014, pp. 359–368. ISBN: 978-1-4503-2698-8. DOI: 10.1145/2554797.2554830.
- [3] László Babai. “On the isomorphism problem”. Manuscript. 1977.

- [4] Leonard Berman and Juris Hartmanis. “On Isomorphisms and Density of NP and Other Complete Sets”. In: *SIAM Journal on Computing* 6.2 (1977), pp. 305–322. DOI: 10.1137/0206023.
- [5] Kellogg S. Booth and Charles J. Colbourn. *Problems Polynomially Equivalent to Graph Isomorphism*. Tech. rep. CS-77-04. University of Waterloo, June 1979.
- [6] Samuel R. Buss et al. “Strong Isomorphism Reductions in Complexity Theory”. In: *The Journal of Symbolic Logic* 76 (4 2011), pp. 1381–1402. DOI: 10.2178/jsl/1318338855.
- [7] Raghavendra Rao B.V. and Jayalal Sarma M.N. “Isomorphism testing of read-once functions and polynomials”. In: *Foundations of Software Technology and Theoretical Computer Science*. Dec. 2011. DOI: 10.4230/LIPIcs.FSTTCS.2011.115.
- [8] Samuel Coskey, Joel David Hamkins, and Russell Miller. “The hierarchy of equivalence relations on the natural numbers under computable reducibility”. In: *Computability* 1.1 (2012), pp. 15–38. DOI: 10.3233/COM-2012-004.
- [9] Rod Downey and Lance Fortnow. “Uniformly Hard Languages”. In: *Theoretical Computer Science* 298.2 (2003), pp. 303–315.
- [10] Ekaterina Fokina, Sy Friedman, and André Nies. “Equivalence Relations That Are  $\Sigma_3^0$  Complete for Computable Reducibility”. In: *Logic, Language, Information and Computation*. Ed. by Luke Ong and Ruy de Queiroz. Vol. 7456. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 26–33. ISBN: 978-3-642-32620-2. DOI: 10.1007/978-3-642-32621-9\_2.
- [11] Ekaterina B. Fokina and Sy-David Friedman. “On  $\Sigma_1^1$  equivalence relations over the natural numbers”. In: *Mathematical Logic Quarterly* 58.1-2 (2012), pp. 113–124. ISSN: 1521-3870. DOI: 10.1002/malq.201020063.
- [12] Lance Fortnow and Joshua A. Grochow. “Complexity classes of equivalence problems revisited”. In: *Information and Computation* 209.4 (2011), pp. 748–763. ISSN: 0890-5401. DOI: 10.1016/j.ic.2011.01.006.
- [13] Su Gao and Peter Gerdes. “Computably Enumerable Equivalence Relations”. In: *Studia Logica* 67.1 (2001), pp. 27–59. ISSN: 0039-3215. DOI: 10.1023/A:1010521410739.
- [14] Su Gao and Caleb Ziegler. “On Polynomial-Time Relation Reducibility”. 2014. URL: <http://www.math.unt.edu/~sgao/pub/paper46.html>. To appear.
- [15] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. New York, NY: W. H. Freeman and Company, 1979.
- [16] Egor Ianovski et al. “Complexity of equivalence relations and preorders from computability theory”. In: *arXiv* (2013). URL: <http://arxiv.org/abs/1302.0580v2>. Preprint.

- [17] Luděk Kučera. “Theory of categories and negative results in computational complexity”. Manuscript. 1976.
- [18] Johannes Köbler, Uwe Schöning, and Jacobo Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Boston: Birkhäuser, 1993. ISBN: 9780817636807.
- [19] Richard E. Ladner. “On the Structure of Polynomial Time Reducibility”. In: *Journal of the ACM* 22 (1 Jan. 1975), pp. 155–171. ISSN: 0004-5411. DOI: 10.1145/321864.321877.
- [20] Russell Miller and Keng Meng Ng. “Finitary reducibilities on equivalence relations”. In: *arXiv* (2014). URL: <http://arxiv.org/abs/1406.3646>. Preprint.
- [21] Mark V. Sapir, Jean-Camille Birget, and Eliyahu Rips. “Isoperimetric and Isodiametric Functions of Groups”. In: *Annals of Mathematics*. Second Series 156.2 (2002), pp. 345–466. ISSN: 0003486X. DOI: 10.2307/3597195.
- [22] John E. Savage. *Models of Computation: Exploring the Power of Computing*. Addison-Wesley, 1998.
- [23] Uwe Schöning. “Graph Isomorphism is in the Low Hierarchy”. In: *Proceedings of the 4th Annual Symposium on Theoretical Aspects of Computer Science*. STACS ’87. London, UK: Springer-Verlag, 1987, pp. 114–124. ISBN: 3-540-17219-X. DOI: 10.1007/BFb0039590.
- [24] V. N. Zemlyachenko, N. M. Korneenko, and R. I. Tyshkevich. “Graph isomorphism problem”. In: *Journal of Soviet Mathematics* 29.4 (May 1985), pp. 1426–1481. DOI: 10.1007/BF02104746.